Individual User Interfaces and Model-based User Interface Software Tools

by

Egbert Schlungbaum

GIT-GVU-96-28
November 1996

Graphics, Visualization & Usability Center

Georgia Institute of Technology

Atlanta, GA 30332-0280

Individual User Interfaces and Model-based User Interface Software Tools

Egbert Schlungbaum[1]

Department of Computer Science
University of Rostock
D-18051 Rostock
Egbert.Schlungbaum@informatik.uni-rostock.de

GVU Center
Georgia Institute of Technology
Atlanta, GA 30332-0280
eggi@cc.gatech.edu

Abstract

Currently, most of model-based user interface software tools use task, application, and presentation models to generate the running user interface. The point of this paper is to use an additional user model to create individual user interfaces. For it, individual user interfaces and model-based tools are analyzed briefly to define the starting point for this research work.

The viability of this approach is discussed with an example using the TADEUS environment. Furthermore, some ideas are presented to extend the MASTERMIND system.

Keywords

Model-based User Interface Software Tools, Model-based User Interface Development, Explicit User Model, MASTERMIND, TADEUS

Acknowledgments

This research is accomplished at Georgia Tech's GVU Center and is supported by funding of the German Academic Exchange Service (Post-Doc fellowship). The author would like to thank Spencer Rugaber, and Kurt Stirewalt for their helpful comments.

[1]accepted short paper for the International Workshop on Intelligent User Interfaces IUI'97

Introduction

When developing an interactive system the developer spends a lot of work and time designing the graphical user interface. This task is error-prone and expensive, and it must meet high quality requirements. For example, the "EEC 90/270 Directive" of the European Council [3] requires that the software (user interface) must be suitable for the task, that the software must be easy to use and adaptable to the operator's level of knowledge or experience. This implies we must design individual user interfaces.

In order to fulfill the high level quality requirements, to reduce time and costs, and to make the task easier the user interface designer needs support from sophisticated development tools. There are several higher level tools to support user interface development, e.g. User Interface Management Systems, Interface Builders, User Interface Development Environments which are built on the top of user interface toolkits. B. Myers [8] has introduced a classification of the mentioned tools: language-based tools, interactive graphical specification tools, and model-based generation tools. Especially the language-based and the interactive graphical specification tools support the specification of either the user interface dynamic behavior or the user interface layout in an easy way but mostly not both parts at one time. These tools support rather the user interface implementation as their task-oriented and user-centered design. One goal of model-based generation tools is to meet these limitations.

This short paper presents some ideas on the extension of model-based user interface software tools for creating individual user interfaces. For it, individual user interfaces are classified. The focus here is on individualization during design time. Furthermore, a short summary of the current state of model-based tools is presented in order to define a possibility of developing individual user interfaces by means of model-based tools. Then, a small example is presented to demonstrate the generation of adapted user interfaces in TADEUS. Finally, some ideas are offered to show the benefit of an explicit user model in MASTERMIND.

Individual User Interfaces

If one interactive application presents itself with different user interfaces to different end users or groups of end users, we will call these *individual user interfaces*. Their need is founded in different user characteristics and preferences, in different tasks users have to

perform with one interactive system. There are mainly two dimensions of individual user interfaces: the characteristics in which they differ and the time when the user interface becomes individual.

Individual user interfaces can differ in available application services (functionality) or presentation layout and dynamic behavior (user interface characteristics). There are three different times when the user interface adapts itself or is adapted to the end user:

- *adapted user interface* - the user interface is adapted to the end user at design time.
- *adaptable user interface* - the end user himself may change (e.g., adapt) the functionality and/or some characteristics of the user interface.
- *adaptive user interface* - the user interface changes its characteristics (presentation layout as well as dynamic behavior and availability of application services) dynamically at run time according to the end user's behavior.

The area of individual user interfaces has attracted a high degree of interest over the last decade. Schneider-Hufschmidt et.al. [15] present a coherent and comprehensive overview of the research of adaptive user interfaces including a state-of-the-art report [2]. A comparative investigation of adaptable and adaptive systems was given by Fischer [4]. This short paper is focused on adapted user interfaces.

It is widely accepted that a system (user interface) which is intended to adapt to users and their tasks needs to model those aspects in some way. Model-based user interface tools can include explicit task and user models.

Model-Based User Interface Software Tools

The approach of model-based user interface development offers a high potential for creating integrated user interface development environments and supporting the whole user interface life cycle. Several model-based user interface software tools have been built, e.g. UIDE [5], HUMANOID [17], MECANO [10], ADEPT [6], TRIDENT [1].

The key idea of model-based generation tools is that all information about user interface characteristics that is required for the user interface development is explicitly represented in declarative models. As shown in Figure 1, many tools support editing and manipulating these models. They allow a comprehensive support for design and implementation. The working

user interface is generated by means of the runtime system which executes the specified models.

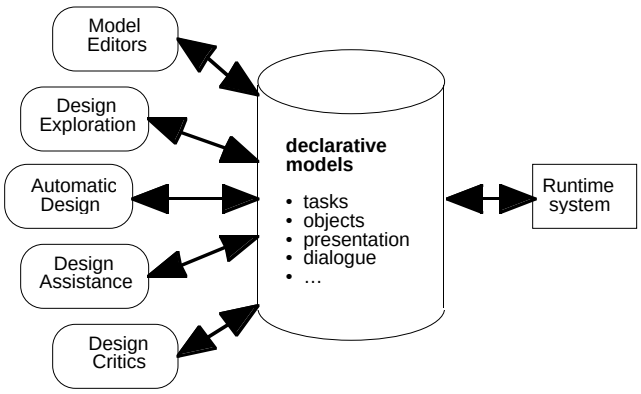

Fig. 1 Model-based approach to user interface development

Different kinds of declarative models are reported for use in model-based tools [14], e.g. task, application, dialogue, presentation, behavior, platform, user, and workplace models. They are used in different ways. The first five are mainly used in the model-based environments mentioned above. The use of an explicit user model was suggested in the context of ADEPT only [7]. Neither an explicit platform nor an explicit workplace model is used in any of the model-based tools.

In order to generate individual user interfaces whether at design time or at run time an explicit user model is required to store the end user's individual information.

User model and model-based user interface development

Model-based user interfaces are developed by using specialized design-time tools to build and refine the declarative models. User interface developers (the end users of model-based environments) specify *what* features the interface should have, rather than write programs that specify *how* to make the computer exhibit the desired behavior. Currently, most model-based tools use task, application, and presentation models to generate the running user interface. This makes the generation of adapted user interfaces difficult because developers have to design different task models or presentation models.

The more natural way is to design universal task, application, and presentation models for one interactive application and to use an explicit user model and its relations to the other declarative models which represent the necessary information to make the generated interface individual.

According to several user model classifications an *embedded* and *stereotypical user model* [2,7] is suggested for use in model-based tools during design time. Stereotypical user models range between canonical (a single model for all users [12]) and individual models. They model groups of users which have some characteristics in common. These groups will be called *roles*.

User characteristics can be classified as application independent and application dependent. Application independent characteristics include preferences, capabilities, psycho-motor skills, etc. Application dependent characteristics include goals, knowledge of system and application, etc [2].

In order to use the user model at design time (to create adapted user interfaces) it must be acquired as result of the comprehensive requirements analysis. The user interface designer has to create this user model. On the other hand, if adaptive user interfaces are the goal, the user model is built during the end user's interaction with the user interface.

Generating individual user interfaces in TADEUS

TADEUS uses task, problem-domain (application), and dialogue models to generate user interfaces [14]. In contrast to tools like UIDE or MASTERMIND which use their own runtime system, TADEUS generates the desired user interface for existing UIMS by means of a user interface description file. With it, generating adapted user interfaces in TADEUS means to generate different user interface description files, one for each role.

The TADEUS user model describes roles hierarchically by means of task independent and task dependent attributes. One role can use an interactive system to carry out several different tasks. In this case, there are some properties such as level of experience using interactive systems, preferences using input devices which have the same value for one role for all different tasks - the task independent *role attributes*. On the other hand, each role must carry out specific tasks. This is specified by means of a *usage relation* between a role specified in the user model and a task specified in the task model and means the role has to

carry out this task. The usage relation has some attributes (the task dependent *usage attributes* such as frequency of use, preferred input device) whose value is only valid when the task is performed [13]. The following example demonstrates how this user model helps to generate adapted user interfaces in TADEUS.

The example describes a task of a banking system handling money transfers. The German bank customer gives a money transfer order using a paper blank. In order to perform the transfer bank clerks must record the data from the blank and check the recorded data. Due to a high security of this procedure there are three clerks with different tasks involved:

- *clerk A* only records new data into empty forms
- *clerk B* checks data forms recorded by clerk A, but cannot record new data or change all fields of one form
- *clerk C* decontrols the data recorded by clerk A and checked by clerk B, but cannot modify the data forms

Only some essential parts of the specification and one possible generation result are shown in Figures 2 and 3 due to the limited space[2]. The *Handling money transfers* task is a sequence of three subtasks. Each of them is carried out by one role. All these subtasks have the same decomposition-pattern: first the user has to open the related object (this enables the other subtasks), then he can finish handling the first and open the second object or finish the subtask. An object specified in the application model is assigned to all tasks but with different access values: read-write or read-only. Only one usage attribute is specified for each role. The dialogue model is omitted. Figure 3 shows the different forms of one design solution for the clerks A and C which are automatically generated from the declarative models. In this case, one universal task and one application model extended with the (currently simple) user model were used to generate the different layouts.

[2]The whole model-based specification and generation results are available at the authors WWW homepage http://www.informatik.uni-rostock.de/~schlung/IUI97/.

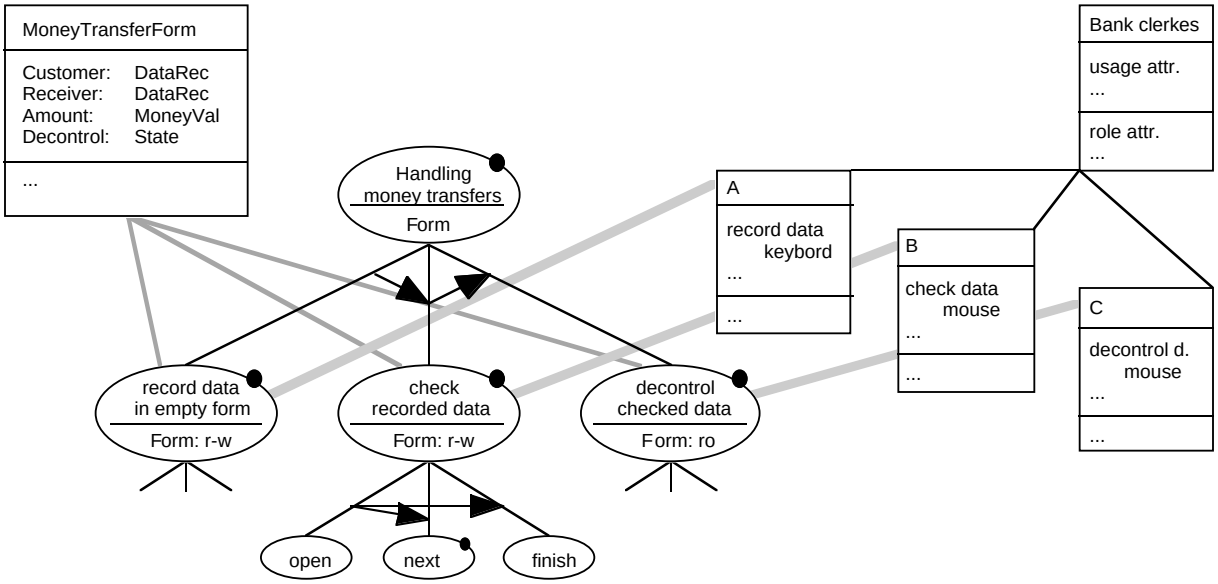

Fig. 2 Task, application, user model and their relations (simplified)

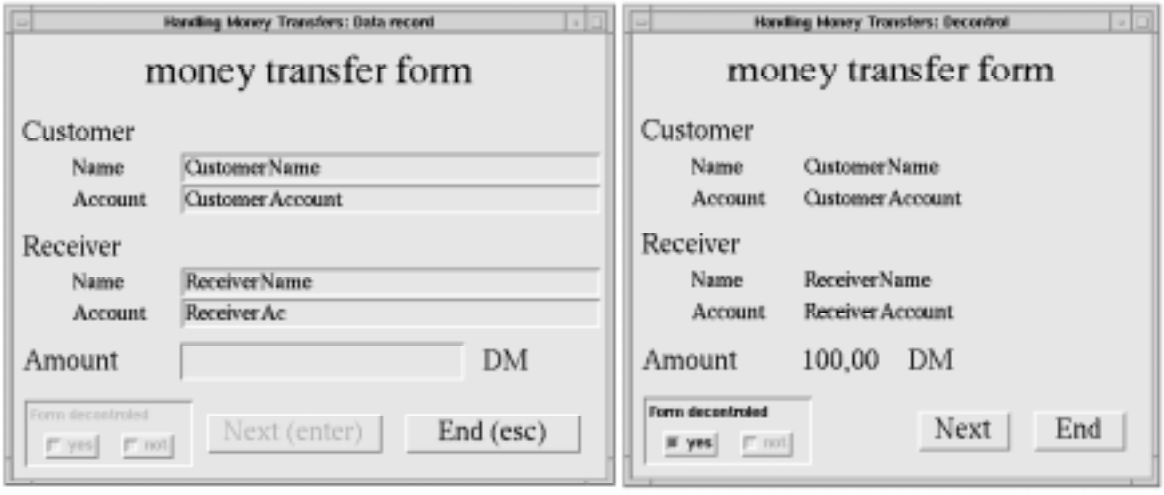

Fig. 3 Adapted forms for clerk A (left) and C (right)

Individual user interfaces in Mastermind

Currently, the MASTERMIND system includes a task, presentation, and application models [18]. Unlike the TADEUS system, in MASTERMIND the running user interface is directly generated from the declarative models thereby preserving the application semantics from design time to run time.

The user interface developer using MASTERMIND could apply a *stereotypical user model* like in ADEPT or TADEUS in different ways (e.g., selecting of appropriate to end user's level of knowledge interaction objects, selecting of presentation parameters, defining of task ordering).

But the architecture of MASTERMIND is more suited to use an explicit user model at run time. An *individual user model* can be used to create adaptive user interfaces as envisioned in UIDE [16]. Unlike UIDE where a narrow user model was integrated into the application model, in MASTERMIND the user model will be an independent part of the declarative models.

Related work

The issue of user modeling to assist the model-based user interface development is investigated in other projects too. As mentioned above, ADEPT [6] is the only model-based tool that includes an explicit user model. It was used to constrain the set of possible design options in the user interface design space. In TRIDENT [1] some user characteristics were directly included into the task model. This leads to a higher modeling effort if one user has to carry out different tasks. MECANO [11] offers a meta level description of a user model, but the presented example does not include a user model.

Conclusion

The first step of a research to extend model-based tools with an explicit user model to support the development of individual user interfaces is presented in this short paper. The current state of model-based tools was reviewed briefly. The viability of this approach was demonstrated with a small example on automatic generating of adapted user interfaces using the TADEUS system. Some ideas were offered to use an explicit user model inside the

MASTERMIND system. The extension of model-based tools with an explicit user model is an issue of current research and first results are expected in the near future.

References

1. Bodart, F., et.al. A Model-Based Approach to Presentation: A Continuum from Task Analysis to Prototype. In [9], 77-94.

2. Dietrich, H., et.al. State of the Art in Adaptive User Interfaces. In [15], 3-48.

3. RICHTLINIE DES RATES vom 29. Mai 1990 über die Mindestvorschriften bezüglich der Sicherheit und des Gesundheitsschutzes bei der Arbeit an Bildschirmgeräten (90/270/EWG). Amtsblatt der EG, Nr. L156, 14-18. (reprinted in SIGCHI Bulletin, Vol. 27, No. 2, April 1995, 18-21).

4. Fischer, G. Shared Knowledge in Cooperative Problem-Solving Systems - Integrating Adaptive and Adaptable Components. In [15], 49-68.

5. Foley, J.D. and Sukaviriya, P. History, Results, and Bibliography of the User Interface Design Environment (UIDE), an Early Model-based System for User Interface Design and Implementation. In [9], 3-14.

6. Johnson, P., et.al. Rapid Prototyping of User Interfaces Driven by Task Models. In Scenario-Based Design: Envisioning Work and Technology in System Development, J. Carroll (ed.), J. Wiley & Sons, London, 1995, 209-246.

7. Kelly, C. and Colgan, L. User Modelling and User Interface Design. In Proc. British HCI'92 (York UK, Sept. 1992), Cambridge University Press, 227-239.

8. Myers, B. A. User Interface Software Tools. In ACM Transactions on Computer-Human Interaction, Vol. 2, No. 1, March 1995, 64-103.

9. Paterno, F. (ed.) Design, Specification and Verification of Interactive Systems '94, Springer, Berlin, 1995.

10. Puerta, A., et.al. Beyond Data Models for Automated User Interface Generation.
In Proc. British HCI'94 (Glasgow UK, August 1994), Cambridge University Press, 353-366.

11. Puerta, A. The MECANO Project: Comprehensive and Integrated Support for Model-based Interface Development.
In Proc. CADUI'96 (Namur B, Juni 1996), Namur University Press, 19-37.

12. Rich, E. Users are individuals: individualizing user models.
In Int. J. Man-Mach Stud 18, 1983, 199-214.

13. Schlungbaum, E. and Elwert, T. TADEUS - a model-based approach to the development of Interactive Software Systems.
In Rostocker Informatik Berichte 17, 1995, 93-104,
(http://www.icg.informatik.uni-rostock.de/~schlung/TADEUS).

14. Schlungbaum, E. and Elwert, T. Automatic user interface generation from declarative models.
In Proc. CADUI'96 (Namur B, Juni 1996), Namur University Press, 3-18.

15. Schneider-Hufschmidt, M., et.al. Adaptive User Interfaces: Principles and Practice.
North-Holland, Amsterdam, 1993.

16. Sukaviriya, P. and Foley, J. Supporting Adaptive Interfaces in a Knowledge-based User Interface Environment.
In Proc. IWIUI'93 (Orlando FL, January 1993), ACM Press, 107-114.

17. Szekely, P., et.al. Beyond Interface Builders: Model-Based Interface Tools.
In Proc. INTERCHI'93 (Amsterdam, April 1993), ACM Press, 383-390.

18. Szekely, P. et.al. Declarative interface models for user interface construction tools: the MASTERMIND approach.
In Proc. EHCI'95 (Grand Targhee Resort, August 1995).